Scraping Relevant Images from Web Pages without Download

ERDINÇ UZUN, Tekirdağ Namık Kemal University, Turkey

Automatically scraping relevant images from web pages is an error-prone and time-consuming task, leading experts to prefer manually preparing extraction patterns for a website. Existing web scraping tools are built on these patterns. However, this manual approach is laborious and requires specialized knowledge. Automatic extraction approaches, while a potential solution, require large training datasets and numerous features, including width, height, pixels, and file size, that can be difficult and time-consuming to obtain. To address these challenges, we propose a semi-automatic approach that does not require an expert, utilizes small training datasets, and has a low error rate while saving time and storage. Our approach involves clustering web pages from a website and suggesting several pages for a non-expert to annotate relevant images. The approach then uses these annotations to construct a learning model based on textual data from the HTML elements. In the experiments, we used a dataset of 635,015 images from 200 news websites, each containing 100 pages, with 22,632 relevant images. When comparing several machine learning methods for both automatic approaches and our proposed approach, the AdaBoost method yields the best performance results. When using automatic extraction approaches, the best f-Measure that can be achieved is 0.805 with a learning model constructed from a large training dataset consisting of 120 websites (12,000 web pages). In contrast, our approach achieved an average f-Measure of 0.958 for 200 websites with only six web pages annotated per website. This means that a non-expert only needs to examine 1,200 web pages to determine the relevant images for 200 websites. Our approach also saves time and storage space by not requiring the download of images and can be easily integrated into currently available web scraping tools, because it is based on textual data.

CCS Concepts: • **Information systems** → **Data extraction and integration**; **Site wrapping** • **Software and its engineering** → *Software design engineering*;

Additional Key Words and Phrases: Web mining, relevant images, web data extraction, crawler design and evaluation

ACM Reference format:
Erdinç Uzun. 2023. Scraping Relevant Images from Web Pages without Download. *ACM Trans. Web* 18, 1, Article 1 (October 2023), 27 pages.
https://doi.org/10.1145/3616849

1 INTRODUCTION

Web pages provide a wide range of data that can be used for a variety of purposes, such as text searching, text classification, image searching, and so on. The process of extracting this data for a specific purpose is known as "web scraping" or "web data extraction." In this issue, the researchers

Author's address: E. Uzun, Tekirdağ Namık Kemal University, Computer Engineering Department, Çorlu Faculty of Engineering, Çorlu, Tekirdağ, Turkey, 59860; email: erdincuzun@nku.edu.tr.

1559-1131/2023/10-ART1 $15.00
https://doi.org/10.1145/3616849

focused mostly on automatically scraping textual data such as titles [27], main content [2, 25, 46], reviews [5, 7, 42, 50], tables [15, 16, 20, 49], main regions [12], and data records [9]. Some studies use both image and textual data for fake news detection [43, 47] and sentiment analysis [51]. However, eliminating irrelevant images from web pages and extracting relevant images is a neglected issue in the literature. Relevant images are among the most crucial data, providing visual information about a web page's content to the web user. This study proposes a semi-automatic approach[1] to scrape relevant images, which improves the performance both in terms of accuracy and computation time.

A web scraper sends a request to a server to retrieve a web page to scrape images. As a response, the server sends the HTML text. The scraper then renders this text and sends other requests to get extra files, including images, Javascript, and **Cascading Style Sheets (CSS)**. That is, the scraper continues to send other requests to the server after the first request. Unlike prior approaches, which required multiple requests to retrieve the relevant image, the proposed approach in this study can extract the source of the relevant image on the first request. In other words, this approach does not necessitate the downloading of images and accelerates the scraping process.

Web scraping tools[2] typically rely on extraction patterns manually prepared by experts from textual data. This means that an expert must examine multiple web pages from a website and identify the appropriate extraction patterns. This process can be both challenging and time-consuming, and the amount of time required can vary depending on the expertise level of the individual involved. Therefore, automatic approaches can provide a potential solution to the challenges associated with manual web scraping tools. These approaches can be divided into two sub-approaches: supervised and unsupervised approaches. In both approaches, the features play a key role. Some features can be obtained from the attributes of HTML elements, such as the text of image elements. However, when it comes to relevant image extraction tasks, some features [45] such as height, width, format, size, and color composition that can be obtained from the image file are more important. Moreover, some features [41] can be collected during the crawling process. All features require extra time for automatic approaches. In unsupervised approaches, an expert searches for a suitable feature and its values by analyzing relevant and irrelevant image features. However, this can be a challenging and time-consuming task that requires examination of many websites and pages. In supervised approaches, a learning model is obtained using machine learning methods with a training dataset. To obtain successful learning models, the selection of features and amount of training data are crucial. However, preparing training data can be a challenging task, and obtaining features can increase the extraction time. In this study, we propose a semi-automatic approach, called **TC (Text-based Classification)**, which combines the strengths of existing approaches. Our approach utilizes textual data as a feature, which is a key component of manual approaches. The clustering step, as in an unsupervised approach, is employed to construct a training dataset, whereas machine learning methods, as in a supervised approach, are used to train a learning model based on features extracted from textual data. To the best of our knowledge, this is the first study that relies exclusively on textual data for the extraction of relevant images. However, due to the wide variation in textual data across websites, our approach generates a custom model for each individual website, similar to the approaches employed in web scraping tools. Therefore, it can be easily integrated into existing web scraping tools,[3] thereby eliminating the need for web experts for these tools.

The preparation of the training dataset for the supervised and TC approaches is one of the most important tasks. In this study, the training dataset included features obtained from the relevant

[1]Our approach and experiments are implemented in Python, and the codes are publicly available on the GitHub repository: https://github.com/erdincuzun/Relevant-Image-Scraping

[2]There are several open-source tools such as Scrapy (scrapy.org), Web-Harvest (web-harvest.sourceforge.net), MechanicalSoup (mechanicalsoup.readthedocs.io), Apache Nutch (nutch.apache.org), pyspider (docs.pyspider.org), StormCrawler (stormcrawler.net), and so on, for this task.

and irrelevant images of many web pages. The number of relevant images on a web page greatly outweighs the number of irrelevant images. As an outcome, an imbalanced dataset was created for this task. To construct an accurate learning model, this study investigated the impact of different machine learning methods and ensemble learning models [26] on this imbalanced dataset. In addition, choosing a varied and particular number of web pages is a crucial issue, as our approach is focused on one website when creating this imbalanced dataset. Therefore, our approach has a clustering step that groups web pages using **URL (Uniform Resource Locator)** data. Manica et al. [28] used a similar approach that combines URL and HTML features to discover entities on on web pages. In contrast, we only collected URLs from several web pages and utilized them for clustering. As far as we know, our proposed clustering step is a novel contribution to reducing the size of the training dataset. A web page has different web layouts, such as home pages, link pages, content pages, gallery pages, video pages, and more, for a given website. In this clustering step, the desired number of web pages is randomly selected from each layout to reduce the risk of overfitting that may occur when using commonly used layouts. Moreover, this step aims to reduce the amount of data markup required for users to annotate the training data. We elaborate on the implications of this step in Sections 4.1 and 5.5, respectively.

The main contributions of this study are as follows:

- A novel approach leverages textual data from HTML elements to perform semi-automatic extraction of relevant images, which can be easily integrated into existing web extraction tools that rely on manual extraction approaches.
- The proposed approach requires minimal features and training data, as annotating a small number of relevant images on a few web pages is sufficient to construct an accurate learning model for a website.
- Unlike other approaches that require downloading images and extracting their features, our approach only uses textual data to construct a learning model. This significantly reduces the computation time and storage space required for data preparation.
- This approach has significantly improved the performance results, and its usability is not limited to image extraction but can also be easily extended to other web data extraction tasks by annotating a small amount of data.
- Our approach's clustering step simplifies the process of creating training data for a website.
- The proposed approach eliminates the need for an expert user to inspect image elements or extract features, making it usable for non-experts.

The rest of the study is organized as follows: The second section gives detailed information about previous studies on web data extraction. The third section introduces the structure of news websites; this section also contains information about existing approaches for extracting relevant images from web pages. The fourth section covers our approach. The fifth section presents the experimental results of these approaches. The last section is devoted to the conclusions and future studies.

2 RELATED STUDIES

Data extraction from a web page can be performed manually or automatically using different approaches. Manual extraction approaches depend on extraction patterns prepared by an expert user. Automatic extraction approaches employ a function or model that does not necessitate the use of an expert.

[3]Our proposed approach has been integrated into the Scrapy Framework, which is a popular and widely used open-source framework in the Python community. (https://github.com/erdincuzun/Relevant-Image-Scraping/tree/main/Scrapy%20with%20TC).

A well-known technique in manual extraction approaches is that the expert user prepares regular expressions [31] that are search patterns to extract data from text. However, preparing these expressions is a monotonous and error-prone operation [13]. Therefore, web developers recommend simple techniques such as CSS selectors and **XML Path Language (XPath)** expressions for accessing HTML elements within web pages. These techniques are based on the **Document Object Model (DOM)** tree. In addition, many libraries in different programming languages developed for web operations support these techniques [35]. Uzun et al. [37] compare three different .Net libraries. They use XPath expressions. Uzun et al. [39] employ three different Python libraries and regular expressions. In Section 3 of our study, the challenges of preparing these expressions are given through some examples of accessing relevant images.

To eliminate the need for an expert user, researchers [24] focus on automatic extraction approaches to obtain the desired data. These approaches can be divided into unsupervised and supervised ones.

Unsupervised approaches use tree-based methods to find the desired element in the DOM tree. Tree edit distance matching [30] is a well-known technique in the literature. This technique has many variants [11, 21, 48] to improve the performance of this task. The functions of these approaches are strictly related to the structure of a website. In this case, supervised approaches can be considered as a solution.

Supervised approaches aim to scrape the desired data from different web pages effectively. These approaches rely on training datasets that are well prepared for domain-specific needs. The first examples based on a training dataset are SoftMealy [18] and STALKER [29]. Bar-Yossef and Rajagopalan [4] propose a practical solution for data extraction based on counting frequent features, including relevant links, the number of pages in a mixture, and lexical affinity. Kohlschütter et al. [22] propose many features, as called "shallow text features," including average word length, average sentence length, the ratio of the number of words that start with uppercase compared to the total number of words, link density, the number of tokens, and so on, for improving the performance of this task. Vogels et al. [44] introduce a model-based sequence labeling for classifying text blocks. Schäfer [32] presents an algorithm based on languages, including English, French, German, and Swedish. Aslam et al. [3] develop an algorithm named Web-AM to remove noise from web pages.

There are few studies in the literature on predicting relevant images. Helfman and Hollan [17] score images by using heuristic measures for a web page. Bhardwaj and Mangat [6] propose a simple technique in which the image size is greater than 120,000 pixels (more than 300 px * 400 px) for this task. This scoring technique is useful for selecting a single image that represents a group of images on a web page. Gali et al. [14] represent a heuristic score obtained from image size, aspect ratio, alt tag, title tag, image path, and image format. In the last study [19], an accuracy of 0.690 is obtained on a small dataset. All of these studies are unsupervised methods. The first supervised approach for this task is proposed by Reference [45]. They adjust some parameters to improve the performance of **Support Vector Machines (SVM)** and reach an f-Measure of 0.439. Uzun et al. [41] try several machine learning methods and some novel features obtained from the crawling process. As a result, employing the AdaBoost method [41] leads to a notable enhancement in performance, yielding an f-Measure of 0.822. Agun and Uzun [1] propose an approach to generating regular expressions that can be used to extract relevant images from shopping websites. Their approach has been evaluated on a small dataset of 360 image elements from 10 different websites, and the results indicate that the approach achieved a prediction success rate of 0.980 f-Measure. This study focuses on a semi-automatic approach that exploits the advantages of both manual and automatic approaches.

3 BAKGROUND

First, the fundamental information about websites, web pages, layouts, HTML elements, and image elements is given to introduce the relevant and irrelevant images on a web page. The second subsection covers the manual extraction by using the parent elements and attributes of an HTML element. The last subsection summarizes the automatic extraction by discovering features obtained from HTML elements.

3.1 Problem Definition

A website consists of different web layouts that define its structure. Relevant and irrelevant images are in these layouts that are created with HTML elements. Each layout consists of similar elements that can be used in the extraction process. Figure 1 shows some layouts, such as the homepage, main article, and gallery, and some examples of web pages on a website.

In Figure 1, relevant images are related to the content of the web page, whereas irrelevant images, including advertisements, other links, headers, logos, and so on, are not related to the content of the web page. Each of these images is a different size and appears in various layouts. For example, in the homepage layout of Figure 1, the slider part typically shows images with a little snippet of text, each linked to other web pages. The number of pixels in these images is larger than the number of pixels in the other images. Although these images look like the relevant images, they are not actually relevant. That is, no images are directly related to the content in this layout.

A web page consists of HTML elements, including a starting HTML tag, attributes, an ending HTML tag, and the content between these tags. While HTML tags and attributes are utilized to design a web page, CSS specifies your page's styles. That is, page layouts, colors, and fonts are all determined by CSS. CSS defines the styles of a web page by interacting with HTML elements. An HTML element is an individual component of a web page. For instance, Table 1 indicates two-parent elements and Img (image) elements with their layout and relevant status.

A web scraper renders HTML elements and constructs a DOM tree from a web page to access them. An HTML element can have attributes that provide additional information. In the first example of Table 1, the class attributes in the Div elements specify two and one class name, respectively. The class attribute is mostly used by CSS. The Img element's src attribute is used to determine the URL of the source image. The Img element is directly above the Div element (class name is nd-article), and these two elements are directly above the Div element (class names are row and img-wrapper) in the hierarchy that is called the parent of the element. In Table 1, the A element defines a hyperlink. The most important attribute of the A element is the href attribute, which determines the link's source. The Ul and Li elements define an unordered (bulleted) list. For extracting relevant images from these elements, there are two different approaches: manual extraction and automatic extraction.

3.1.1 Manual Extraction. In manual extraction, a web expert examines the elements of web pages on a website and focuses on determining the appropriate patterns for the extraction process. In making this determination, the expert generates patterns and checks these patterns on many web pages of a website to discover a convenient pattern. If the pattern is not correct, then an expert comes up with new patterns that take into account the parent elements.

In the extraction process, three standards can be used to access an HTML element: CSS selectors, XPath expressions, and regular expressions. For instance, to obtain the source of the Img element in the first example of Table 1:

—CSS Selector: div.row.img-wrapper div.nd-article img
—XPath: div[@class="row img-wrapper"]//div[@class="nd-article"]//img
—<a.*?<div.class="slider_img".*?<img.src="(.*?)"

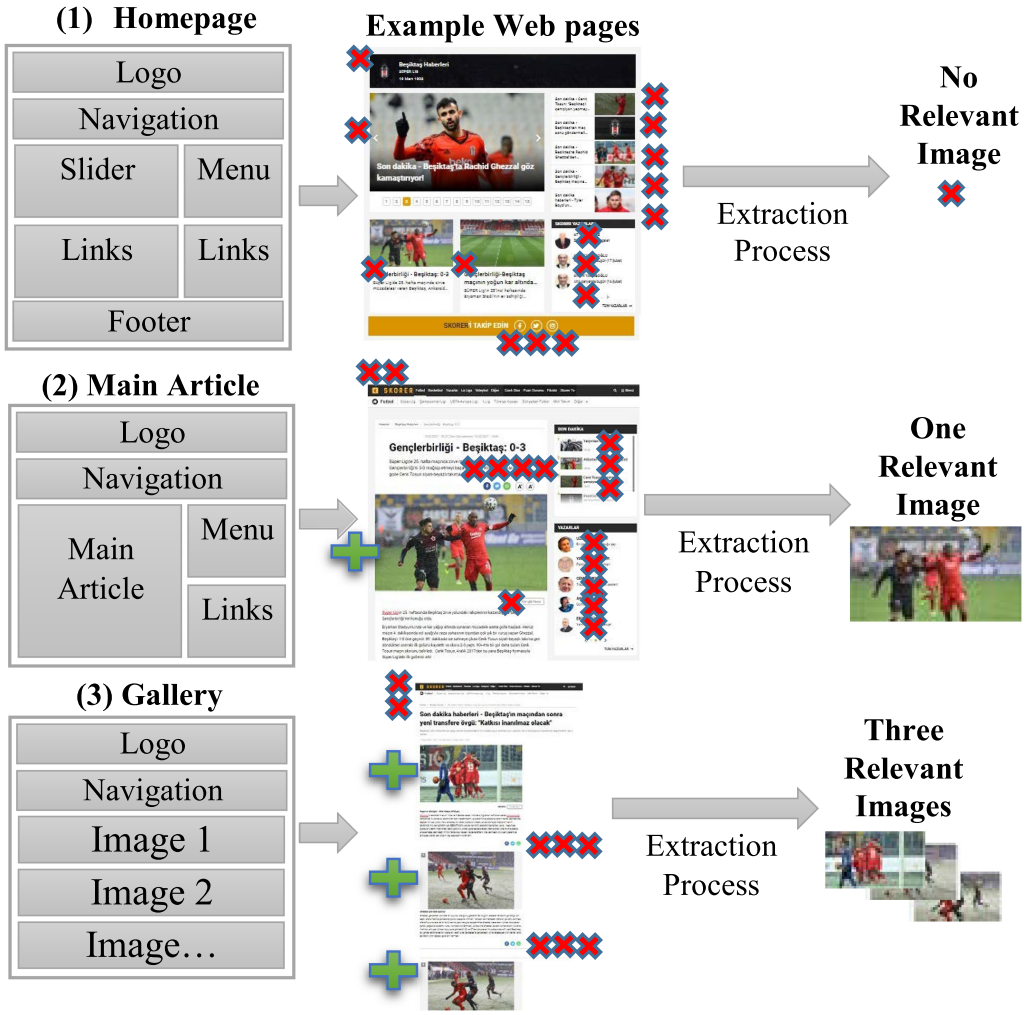

Fig. 1. Relevant images in several layouts in a website. Green (+) and Red (X) represent relevant and irrelevant images, respectively. The homepage layout (1) does not contain relevant images. The main article layout (2) contains one relevant image. But, this layout may have many relevant images or no images at all. The gallery layout (3) contains more images. Also, each of these layouts contains dozens of irrelevant images.

The patterns of CSS and XPath use whitespace and // between elements, respectively. For the class attribute of the div element, "." and [@class="row img-wrapper"] can be expressed, respectively. These extraction approaches need a DOM tree for this task. However, regular expressions do not need this tree but they are more complex for representing patterns for matching text. The dot (.), the question mark (?), the asterisk (*), and regular parentheses are special characters used for string matching. The dot (.) matches any one character. The asterisk (*) is used to match zero or more occurrences. The question mark (?) means to match zero or one instance. The parentheses create your capturing group. This pattern points to the group of the image source data.

These three patterns are sufficient for #1 in Table 1. Although there are relevant images in #2 and #3, the web designer who prepared this layout preferred different attributes. A simple

Table 1. An Example of HTML Elements and Their Relevant Status

#	L	Parent 1, Parent 2, Img (Textual Data)	R
1	2	<img src="0x410/6032d.jpg">	1
2	3	<img src="0x310/90458d.jpg">	1
3	3	<img src="0x300/34578d.jpg">	1
4	1	 <img src="0x410/60336c.jpg"> 	0
5	2	<li class="breaking-news__item"> <img src="91x51/6033c.jpg"> 	0
6	3	<ul class="authors"> <li class="author"> <img src="70x70/5c8bbc.jpg">	0

*L: Layout, R: Relevant.

expression like div.picture (CSS Selector) can be prepared to get the relevant images of #2 and #3. But, in this situation, #4 element is omitted as relevant, although it is irrelevant. The manual extraction approaches become more error-prone due to the complexity of the problems. There are many similar expressions and patterns for different situations in this task. Moreover, the examples in Table 1 are extremely simple and straightforward. The web designer is free to set his elements and attributes. Also, many HTML tags and attributes can be used to create elements/layouts on web pages. The contents and numbers of the elements/layouts are variable on each web page. Moreover, there are big differences between websites and the layouts of websites.

3.1.2 Automatic Extraction. The automatic extraction approaches aim to facilitate this process through a dataset. These approaches can be classified into two categories: unsupervised and supervised. Figure 2 illustrates these approaches.

Unsupervised approaches for extracting relevant images rely on functions created by experts. For example, Bhardwaj and Mangat [6] proposed a function that identifies images with a width greater than 300 pixels and a height greater than 400 pixels as relevant. In contrast, supervised methods require a larger set of features and a significant amount of training data to build an accurate learning model. The process of preparing training data and selecting features is time-consuming. Once these tasks are complete, a machine learning model is constructed using the training data to determine whether an image is relevant or not. Different machine learning methods are evaluated on the dataset to determine the most successful model for the prediction process. In Section 5 of our study, we compare various methods proposed in the literature, and their performance results serve as baselines for the experimental section.

4 TC APPROACH

The TC approach focuses on utilizing textual data, similar to manual extraction approaches, to extract relevant images. Unlike unsupervised approaches that rely on an expert user and supervised approaches that require a significant amount of training data, the TC approach involves a standard web user annotating a few relevant images on a website. The textual data is then used to construct a learning model for the website, consisting of three key steps: clustering, annotation, and machine learning. Figure 3 provides a visual representation of these steps.

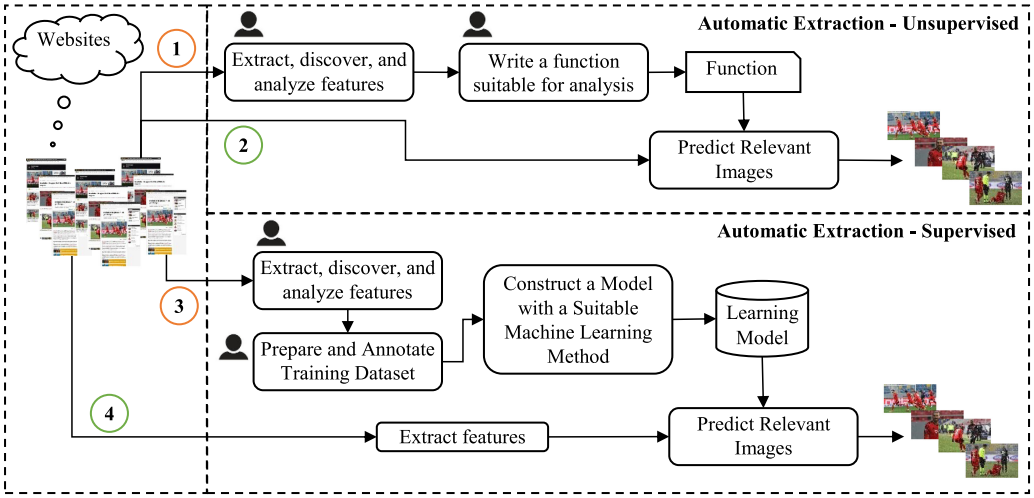

Fig. 2. Two different automatic extraction approaches. In the first step (1) of unsupervised automatic approaches, an expert user makes a deeper analysis to find an appropriate function that can be used for different websites. The second step (2) of this approach utilizes this function for scraping the relevant images. In the first step (3) of supervised automatic approaches, an expert user finds features and applies machine learning methods instead of a deep analysis process. First, a training dataset containing the features is constructed, and a learning model is produced from this training data with machine learning methods. The second stage of this approach (4) extracts features from a web page and then uses the learning model to predict which images are relevant/irrelevant.

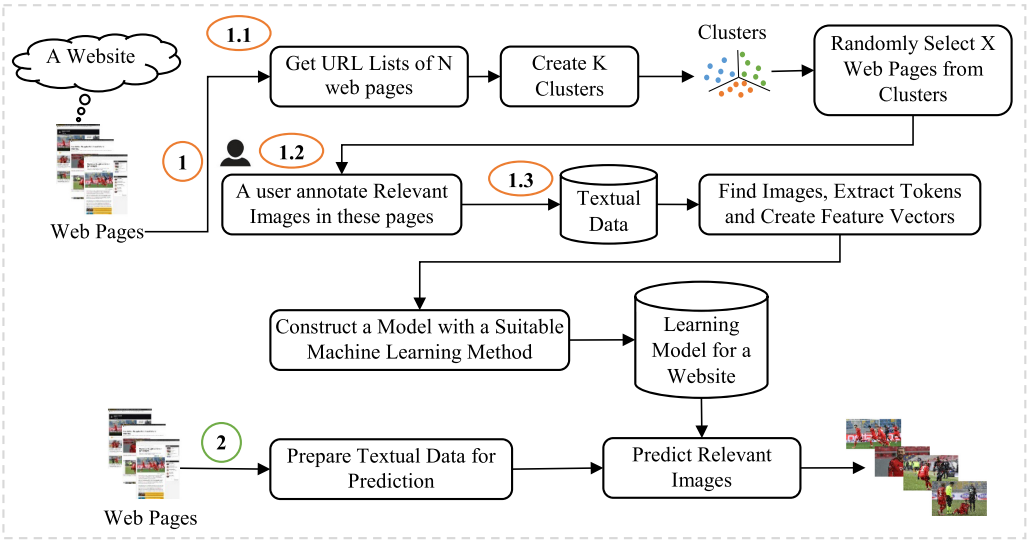

Fig. 3. Text-based Classification (TC) Approach. The first step (1) contains three sub-steps: clustering (1.1), annotation (1.2), and machine learning (1.3). After the learning model is constructed for a website, the image predictions (2) are made on the textual data of an image.

4.1 The Clustering Step

As previously mentioned in Section 3.1, web pages can have diverse layouts. Typically, web pages with the same layout on a website share similar HTML elements. When preparing a training dataset, selecting a range of web pages that represents various layouts is crucial to enhance the model's performance and prevent overfitting. Randomly picking web pages may lead to missing out on certain layout types in a dataset. To overcome this challenge, three techniques can be used to automatically identify various web layouts.

(1) Capturing a screenshot of the web page and using machine learning methods to identify the web layout [34].
(2) Determining the web layout through the application of machine learning or heuristic techniques, following the download and creation of a DOM tree from the textual data of a web page [15].
(3) Using URL data to predict web layout from other web pages collected from multiple web pages of a website.

The first technique involves making URL requests for files such as images, Javascript, and CSS, as well as downloading the entire web page. Capturing a screenshot of the web page and processing it with machine learning methods is resource-intensive. The second method only requires the text of the web page and considers all the texts within the web pages for web layout detection. The third approach, which is time-efficient, does not require downloading the web page and instead uses URL data collected from several web pages to identify web layouts. This technique is more lightweight than other techniques, because it uses the clustering step that is less demanding. In simpler terms, in the first two techniques, the user waits for hundreds of pages to be processed before annotating relevant images. The technique proposed in this study significantly reduces the waiting time compared to the other two techniques. The clustering step, which utilizes URL data, is included in Algorithm 1.

In Algorithm 1, to obtain a sufficient number of $URLs$, several web pages are downloaded. Moreover, the algorithm takes the number of URLs ($trainingSize$) to be included in the training dataset as a parameter, and the exploration of the impact of this number is presented in Section 5. Algorithm 1 is utilized to select the URLs ($selectedURLs$) that the user should choose to annotate relevant images. ***DBSCAN* (Density-based spatial clustering of applications with noise)** [8] function in the algorithm takes the URLs and the output of lev function among these URLs and divides the URLs into c clusters. In lev function, all strings in the URLs are compared with each other, and Levenshtein distance [23] values are calculated for comparisons. This distance is a metric for measuring the difference between two strings. Moreover, there are various distance metrics available, such as **LCS (Longest Common Subsequence)** and Hamming distance. However, when working with URL data, it is observed that operations such as deletion, addition, or substitution are commonly performed. The Levenshtein distance, also known as the edit distance, can support all three operations and is widely used in literature for string similarity measurement. However, other distance metrics do not fully support these operations. Therefore, in this study, we chose to use the Levenshtein distance as the distance metric for clustering URLs. Equation (1) demonstrates how to calculate these distances for the URL list's values.

$$\mathrm{lev}_{x=url_i, y=url_j}(m,n) = \begin{cases} \max(\mathrm{m}, \mathrm{n}) & \text{if } min(m,n), \\ \min \begin{cases} \mathrm{lev}_{x,y}(m-1,n)+1 \\ \mathrm{lev}_{x,y}(m,n-1)+1 \\ \mathrm{lev}_{x,y}(m-1,n-1)+1_{(x_m \neq y_n)} \end{cases} & \text{otherwise} \end{cases} \quad (1)$$

ALGORITHM 1: The Clustering Step of TC

```
Data:
URLs: URLs downloaded from several web pages on a website
trainingSize: desired number of URLs for the annotation step
Result: selectedURLs: recommended URLs for training data
selectedURLs ← [ ];
clusters ← DBSCAN(URLs, lev(URLs));
countClusters ← [ ];
for i ← 0 to Length(clusters) by i++ do
    Append selectedURLs ← Select and Pop a URL from clusters_i;
    Append countClusters ← Length(clusters_i);
    if trainingSize == Length(selectedPages) then
        break;
    end
end
restTrainingSize = trainingSize - Length(clusters);
if restTrainingSize > 0 then
    for i ← 0 to Length(countClusters)) by i++ do
        countClusters_i ← countClusters_i * restTrainingSize/ Length(URLs);
    end
    for i ← 0 to Length(CountClusters) by i++ do
        for j ← 0 to clusters_i by j++ do
            Append selectedURLs ← Select and Pop a URL from clusters_i;
        end
    end
end
```

In $x = url_i$ and $y = url_j$, the two URL values are compared to calculate the distance. In this comparison process, values, where i is greater than j, are taken into account. This prevents the values from being compared twice. m and n are the terminal character positions of values. x_m and y_n refer to the characters of value url_i and url_j at positions m and n, respectively. This definition corresponds directly to the naive recursive implementation. After determining all Levenshtein distances, the number of clusters is determined by using the *DBSCAN* based on these values. As a result, the *DBSCAN* function divides the URLs into *clusters*, where each cluster represents web pages with similar layouts. The value of *clusters* varies, depending on the website. However, selecting only one web page from each cluster can create problems if some websites do not contain relevant image/s on their page/s. To address this issue, the user who creates the training data can request as many pages as needed from these sets. Regarding the time complexity, the second line of the code primarily depends on two factors: the number of character comparisons, which is proportional to the product of the URL lengths ($m * n$), and the size of the training dataset, denoted as $trainingSize$. Consequently, the overall time complexity of this line can be expressed as $O((trainingSize^2/2 - trainingSize) * (m * n))$. Assuming that all parameters take the largest value, the worst-case time complexity of the algorithm is $O(N^4)$. It is crucial to investigate the impact of the training dataset size and the lengths of URLs on the performance of the approach in Section 5.5. By analyzing these factors, the optimal values can be determined to enhance the effectiveness and efficiency of the approach.

Table 2. A Simplified Example for Algorithm 1

URLs, trainingSize: 5					
url_1	news?page=2				
url_2	agenda/example-234324				
url_3	agenda/example-304222				
url_4	agenda/sample-099642				
$url_{n=15}$	...				
Levenshtein Distances (*lev* function)					
	url_1	url_2	url_3	url_4	$url_{n=15}$
url_1		17	17	16	...
url_2			4	8	...
url_3				7	...
url_4					...
$url_{n=15}$	...	...	...	...	...
DBSCAN					
Cluster$_1$: url_1, $url_1 2$					
Cluster$_2$: url_2, url_4, url_8, url_9, url_{10}, url_{13}					
Cluster$_3$: url_3, url_5, url_6, url_7, url_{11}, url_{14}, url_{15}					
First loop in Algorithm 1					
$selectedURLs$: url_{12}, url_{10}, url_5					
$CountClusters$: 2, 6, 7					
Second loop in Algorithm 1					
$restTrainingSize$: $5 - 3 = 2$					
$CountClusters$: 0, 1, 1					
Third loop in Algorithm 1 (Output)					
$selectedURLs$: url_{12}, url_{10}, url_5, url_4, url_{14}					
Random Selection from Clusters for $trainingSize = 5$ where $trainingSize$ is the number of URLs given by user					

The first loop of Algorithm 1 involves selecting a web page from the *clusters* and adding it to the *selectedURLs* list. The number of web pages in each cluster is also stored in the *countClusters*. If the number of URLs (*trainingSize*) requested by the user is equal to the number of clusters, then the selected web pages (*selectedURLs*) are returned as an output. Otherwise, the second loop is initiated. In the second loop, the algorithm calculates the weight of each cluster (*countClusters*) based on the number of URLs it contains. Then, in the final loop, the algorithm randomly selects web pages from each cluster (*clusters*) according to their weight. The execution time of this part of the algorithm is directly proportional to the size of the training dataset, resulting in a time complexity of $O(trainingSize)$. It is worth noting that the most impactful part of the algorithm lies in the DBSCAN line, which performs the actual clustering process. A detailed examination and evaluation of this step will be presented in the Experiments section. Table 2 presents a simplified demonstration for Algorithm 1.

In Table 2, the Levenshtein distances (*lev* function) are calculated for 15 simplified URLs. Based on these values, *DBSCAN* function returns three clusters of related URLs. In the first loop of the algorithm, one URL is randomly selected from each cluster. Subsequently, additional selections are made by considering the number of URLs desired for the training data size (*trainingSize*) and taking into account the weights of the clusters. After completing the clustering step, the algorithm outputs five URLs (*selectedURLs*) for the annotation step.

4.2 The Annotation Step

In the annotation step, a user downloads web pages from the suggested URLs and selects the relevant image/s from these pages. To perform this step, the user is required to view all images on a web page, which necessitates downloading the images. However, in the following steps, it is not necessary to re-download the images. Algorithm 2 outlines the process of how the training dataset is prepared and how the relevant image is annotated for a web page on a website.

ALGORITHM 2: Preparing a Training Dataset and Annotating the Relevant Image

Data:
URLs: N URLs suggested by the clustering step
Images: The image elements extracted from a web page
annotations: The selected image element/s as relevant
textualData: string of Parent 1, Parent 2, and Img. See Table 1
Result: $trainingDataset = [(textualData_i, relevant_i)]$

```
trainingDataset ← [ ];
for i ← 0 to Len(URLs)) by i++ do
    Download WebPages_i from URLs_i;
    Parse Images ← WebPages_i;
    for j ← 0 to Len(Images) by j++ do
        Extract textualData_j ← Images_j;
        Append trainingDataset ← (textualData_j, 0);
    end
    annotations ← user annotates only relevant image/s;
    for k ← 0 to Len(annotations) by k++ do
        i ← Find annotation_k in trainingDataset;
        Update trainingDataset ← (textualData_j − k, relevant_j ← 1)
    end
end
```

Algorithm 2 starts by downloading the web page for each URL specified in the clustering step. It then extracts images from the downloaded web page. For each image, two parent HTML elements are found and the textual data is obtained from these elements. The textual data is composed of the string of the image element and its two parent elements, as given in Table 1. Len function returns the length of a given variable. Append function inserts a given value to the end of the training dataset. Initially, the relevant status of each image is set to 0. Then, the user annotates the relevant image/s for a web page suggested by the clustering step. Finally, all these annotations are searched among the dataset, and the relevant status of the image is assigned as 1. The training dataset is used in the machine learning step of the TC approach to construct a learning model.

The time complexity of Algorithm 2 is $O(trainingSize * (IRI + 2 * RI))$, where IRI represents the number of irrelevant images and RI denotes the number of relevant images within the web page obtained from a given URL. In the worst-case scenario, the algorithm exhibits a time complexity of $O(N^2)$, which is comparable to supervised approaches. However, these supervised approaches typically require a substantial amount of training data, requiring a larger value for $trainingSize$. In contrast, Algorithm 1 aims to minimize the value of $trainingSize$ by leveraging its unsupervised nature. By focusing on relevant images and carefully selecting representative web pages, the approach aims to achieve effective training with a smaller dataset. To determine the optimal value for $trainingSize$ in our approach, comprehensive experiments will be conducted, as described in the Experiments section.

4.3 The Machine Learning Step

To apply machine learning methods, the training data obtained from the annotation step needs to be transformed into feature vectors, which are numerical representations of the data in a two-dimensional vector. Algorithm 3 indicates how to prepare feature vectors.

ALGORITHM 3: Preparing Feature Vectors

Data: *trainingDataset*: obtained from the annotation step
Result: *dictionary*: contains tokens obtained from the training dataset
$dictionary = \{token_1, token_i, \ldots, token_n\}$
featureVectors : stores the frequency of the tokens for all images in the training dataset
featureVectors = [Len(*trainingDataset*)][Len(*dictionary*)]

```
for i ← 0 to Len(trainingDataset) by i++ do
    bagofTokens_i ← Tokenization(Pre−tokenization(textualData_i));
    foreach token in bagofTokens_i do
        if token not in dictionary then
            Append dictionary ← token
        end
    end
end
for i ← 0 to Len(trainingDataset) - 1 by i++ do
    Set all featureVectors[i][Len(dictionary - 1] ← 0;
    foreach token in bagofTokens_i do
        pos = Find the position of token in Dictionary;
        featureVectors[i][pos] ← Count token in bagofTokens_i;
    end
    featureVectors[i][Len(dictionary)] ← trainingDataset.relevant_i;
end
```

The time complexity of preparing vectors in Algorithm 3 can be analyzed in two parts. First, the time complexity of the first loop is determined by the size of the training dataset, denoted as $trainingDatasetSize$, and the length of *tokens* obtained from the textual data, denoted as $tokenSize$. Therefore, the time complexity of this loop is $O(trainingDatasetSize * tokenSize)$. Second, the second loop involves the *Count* function, which operates on the tokens obtained. As a result, the time complexity of the second loop is $O(trainingDatasetSize * tokenSize^2)$. Consequently, the worst-case time complexity of the algorithm is $O(N^3)$, assuming all parameters reach their maximum values. However, our proposed approach offers two distinct advantages over supervised approaches. First, it requires a smaller training dataset, reducing the computational burden. Second, unlike supervised approaches that involve obtaining multiple features with high time complexity, our approach solely relies on the tokenization process. This significantly simplifies and speeds up the feature extraction step. For example, Uzun et al. [41] proposed 30 features that can be obtained from the text of image elements, image files, and crawling processes. In the text of image elements, the algorithm searches for the presence or value of attributes. For example, if the "id" attribute is present in the text, then it is stored as 1, otherwise, it is stored as 0. If the text contains the "width" feature, then the numeric value associated with it is extracted. Image files provide numerical features such as image width, height, and file size. To extract the width feature from an image file, the file is opened and its metadata is parsed. Crawling features are obtained during the crawling process of a website. For example, the cache property indicates whether the source of an image has been previously downloaded. This feature is represented by a numeric value of 0 or 1. The cache feature has been found to improve the outcomes of supervised approaches, but it also raises

processing time, as the number of pages and images increases during crawling. As a result, to extract features from image files for supervised approaches, a web scraper is required to download the images. Moreover, features obtained from the crawling process need extra storage. These processes not only increase the storage requirements but also add extra processing time to the web scraping procedure. In contrast, our approach does not require these processes. Our approach focuses solely on textual data, and we do not search for numerical values within this data. Instead, our algorithm performs tokenization to process the textual data.

The first loop in Algorithm 3 is responsible for creating a dictionary that stores the tokens extracted from the textual data. This dictionary enables efficient lookup operations in the second loop. As a result, the time complexity of the *Find* function in the second loop is $O(1)$, which indicates constant time complexity for finding tokens. In this algorithm, Pre-tokenization function is applied to each textual data item in the training dataset. In this function, "<" and ">" characters are discarded while "/", ".", "?", and ";" characters are replaced with the space character. After this process, the data becomes available for tokenization. Tokenization function is the task of splitting text into smaller tokens. As a result, these functions return tokens named the "bag of tokens" (*bagofTokens* in Algorithm 3) for textual data. The bag of tokens for the first example in Table 1 is:

—div, class=, row, img-wrapper, div, class=, nd-article, img, src=, 0x410, 6032d, jpg

Then, Algorithm 3 checks whether these tokens are in the dictionary. If the token is not in the dictionary, then it is appended to the dictionary. The second loop in Algorithm 3 is utilized for constructing the feature vectors. These vectors contain all the tokens in the dictionary and the relevant status of the image. At the start of the second loop, the relevant status ($featureVectors[i]$[len(*dictionary*)] in Algorithm 3) is set to zero for a given image. Then, Find function returns the position of a given token in the dictionary. Count function calculates the frequency of occurrence of the token in the bag of tokens. That is, these vectors store the count of the times each token appears in each image. Finally, the relevant status of the element is updated at the end of the feature vectors. An example row of the feature vectors for the first example in Table 1 is:

—(div, 2), (class=, 2), (row, 1), (img-wrapper, 1), (nd-article, 1), (img, 1), (src=, 1), (0x410, 1), (6032d, 1), (jpg, 1), (upper, 0), (check-in-view, 0), (picture, 0), (0x310, 0), (90458, 0), (0x300, 0), (34578d, 0), (a, 0), (href, 0) ... (Relevant Status, 1)

After preparing the feature vectors for all images in the training dataset, various machine learning methods can be applied to construct the learning model. The goal of learning models is to determine which tokens are important in predicting relevant and irrelevant images and then build a model on these tokens. In other words, they focus on discovering the most suitable tokens, just like an expert user in manual extraction. Simple notation of the machine learning step:

Learning Model ← *Prepare(trainingDataset for a website, machine learning method),*

where Prepare is a function that first converts the training dataset to the feature vectors and then constructs the learning model for a given machine learning method.

4.4 The Prediction Step

A machine learning model has important parameters, including the training dataset and the machine learning method. Our approach is evaluated on different websites and machine learning methods in the experimentation section. The aim is to make predictions on unseen data and evaluate the prediction accuracy. Our approach's prediction step involves downloading a webpage from a website, but it does not download the image resources on the page. In our approach, the textual data of an image and its two parents are extracted from the web page of a website as input.

Table 3. Statistical Information about the Dataset

	Total / Average ± Std.	
Number of websites	200	
Average size of web pages	244 ± 268 KB	
Number of images (Total)	635,015	
Number of images*	31.846 ± 30.249	
Average size of images	36 ± 241 KB	
Average URL length	94.787 ± 18.467	
Lowest/Longest URL length	25 / 679	
Average Token length	37.429 ± 28.618	
Lowest/Longest Token length	9/2,118	
	Irrelevant (IRI)	**Relevant (RI)**
Total of Images	612,333	22,682
Average number of images*	30.715 ± 30.214	1.285 ± 1.186
Maximum number of images*	264	37
Average size of images	34 ± 242 KB	100 ± 177 KB

KB: KiloByte, Std.: Standard Deviation, *: per web page.

Tokenization is then applied to this input, as described in Algorithm 3. The tokenized data must then be converted into a format suitable for the prediction learning model vector. The prediction model produced for the website is used to make predictions. The notation for this prediction step is:

Prediction Result ← Prediction(textual data of image and its two parents, learning model of a website for a machine learning method).

The prediction function uses a machine learning model to determine if an image on a web page is relevant or irrelevant. This function is applied to all images on the page. The Experiments section evaluates the prediction results of several machine learning methods on multiple websites.

5 EXPERIMENTS

The first subsection presents information about the dataset and the average time results for several operations in approaches. The second subsection summarizes the performance metrics. The third subsection gives brief information about machine learning methods. The fourth subsection indicates the performance results of supervised, unsupervised, and TC approaches. The fifth subsection covers the impact of the clustering process and the size of the training dataset. The sixth subsection explores whether textual data can be used in supervised approaches and the impact of the training dataset size. The last section contains a discussion of the research findings and results.

5.1 Dataset and Time/Storage Costs

To compare the performance results, the dataset[4] used in our study was also employed in the previous study [41]. This dataset contains 100 web pages for each of the 200 news websites, consisting of 58 different countries. That is, it has 20,000 web pages. These web pages have 635,015 images in total, 22,682 of which are relevant images. Tables 3 and 4 summarize some statistical information about the dataset and time/storage costs of manual, unsupervised, supervised, and TC approaches, respectively.

Table 3 presents statistical information that was obtained from analyzing 20,000 web pages from 200 different websites. The average file size of a text-only page is around 244 KB, and each page

[4]The dataset is available for access via the webpage https://adys.nku.edu.tr/Datasets/

Table 4. Statistical Information about Time/Storage Costs of Approaches and Basic Comparisons

		Approaches			
	Average ± Std.	**Manual**	**Unsupervised**	**Supervised**	**TC**
Constructing a DOM tree*	0.039 ± 0.046 s	✓	✓	✓	✓
Downloading images*	0.510 ± 7.063 s	X	✓	✓	X
Preparing image features*	0.008 ± 0.027 s	X	✓	X	X
Preparing 30 features*	0.086 ± 0.116 s	X	X	✓	X
Tokenization*	0.048 ± 0.127 s	X	X	X	✓
Total size of images*	1,147 ± 726 KB	X	✓	✓	X
Requires an expert		+++	+++	++	+
Training size		+	+	+++	+

s: seconds, KB: KiloByte, Std.: Standard Deviation, *: per web page.
✓: Requires processing, X: Processing not required.
+: The level of difficulty.
The experiments were conducted on a computer running Windows 10 (64 Bit) with an Intel Core i7-8850U CPU 1.80 GHz. and 16 GB of RAM. The client bandwidth was 30 Mbps.

typically contains about 32 images. On average, an image file size is 36 KB, but the standard deviation of this value is high at 241 KB. Our clustering step includes an average of 94.787 characters extracted from the URLs we use. Additionally, the textual data within an image typically consists of approximately 37.429 tokens. When considering all of the images in the dataset, only 3.57% of them are relevant images. On average, there is one relevant image per web page, but some pages have as many as 37 relevant images. These pages are typically gallery pages. The maximum number of irrelevant images on a web page is 264. The average file size of a relevant image is 100 KB. When examining the values and standard deviations are given in Table 3, it is seen that the standard deviation is high. This may indicate that the values in the dataset are spread out over a wide range or that there is a significant amount of variability in the data. Table 4 groups and compares the operations used in the approaches.

Table 4 presents the durations of various operations, such as creating a DOM tree, downloading images, preparing image features, generating 30 features, and tokenization, all of which are recognized as time-consuming tasks. The duration of these operations may vary, depending on various factors such as the operating system, CPU speed, amount of available RAM, file size, client and server bandwidth, and so on. When a web scraper requests a web page from the server, it first downloads the HTML text, which is then parsed to create a DOM tree that allows access to the HTML elements. On average, this process takes 0.039 second for a single web page across all approaches. However, the duration varies significantly among different websites, since web pages may vary in HTML text size and the number of elements. As a result, the standard deviation for this operation is as high as 0.046 second.

After obtaining the DOM tree of the web page, the scraper needs to request the server for downloading image, JavaScript, and CSS files. A separate request is sent for each file. Among these operations, downloading images in Table 4 is the most time-consuming one. Furthermore, since there can be a large number of images, this operation may take significantly more time compared to others. However, manual and TC approaches do not require downloading images, resulting in a notable improvement of 0.510 second. This improvement is even higher on some web pages, considering the high standard deviation value of 7.063 seconds. In contrast, unsupervised and supervised approaches in the literature need to download images.

After constructing the DOM tree of a web page, the next step is to extract features from the web page's images. In the manual approach, an expert user identifies patterns within the text data

of the web page without performing any feature extraction. In unsupervised approaches, features such as height, width, and file size of the image file are used. These features can be extracted in 0.008 second for all images on a web page after downloading them. Supervised approaches require many features briefly described in Section 4.3, and it takes an average of 0.086 second to prepare 30 features for images on a web page. In the TC approach, tokenization is performed, and an average of 0.048 second is obtained for all images on a web page. Supervised and unsupervised approaches require image download, which causes significant time loss in the operations. Since the TC approach does not require image download, only two operations, including DOM tree construction and tokenization, are necessary. In other words, the TC approach takes an average of 0.087 second to construct the DOM tree and perform tokenization on a web page, while the supervised and unsupervised approaches require a total of 0.557 and 0.635 second, respectively. In summary, the TC approach has a significant advantage, as it does not need to download images, which is the most time-consuming operation in supervised and unsupervised approaches.

In terms of storage efficiency, the approaches are evaluated based on their requirements for storing images and training datasets. The supervised and unsupervised approaches require image downloads. Table 4 indicates that they necessitate storing around 1 MB of image data for each web page. The manual and unsupervised approaches can work with a smaller amount of training data, depending on the level of expert knowledge. However, supervised approaches require more training data. Our recommended TC approach does not require an expert and has the advantage of not requiring the download of images and requiring less training data, resulting in both time and space savings. The next sections focus on the performance predictions of the approaches and the impact of training dataset size.

5.2 Performance Metrics

In a machine learning study, the most common performance metric is accuracy. However, we encounter the problem known as the imbalanced dataset in the literature, because the number of irrelevant images is too high in our dataset. Accuracy is the ratio of both relevant and irrelevant correct image predictions to all other predictions. For example, if we annotate all images as irrelevant as the simple prediction model, then the accuracy value is 0.96. This value may seem very successful at first glance. However, it has no success in the relevant image prediction. For this reason, accuracy is generally not a very reliable metric. Instead, precision, recall, and f-measure are often used in the literature for performance evaluation. Finally, Log Loss metric is commonly used to evaluate the performance of learning models that make predictions in the form of probability values between 0 and 1. Log Loss measures the discrepancy between the predicted probabilities and the actual class labels. A learning model with a Log Loss value close to 0 is considered to have accurately predicted the class labels, while a higher Log Loss value indicates that the predictions deviate more from the true class labels.

5.3 Machine Learning Methods

Several machine learning methods, including **Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Decision Tree (J48), Random Forests (RF)**, and AdaBoost, have been compared to discover a more successful learning model. SVM is a discriminative machine learning method that supports different kernel functions, including linear, polynomial, and Gaussian **radial basis function (RBF)**. In this study, the RBF kernel was used because it is useful when the data points are not linearly separable. Besides, this kernel has been implemented and proposed by Vyas and Frasincar [45] for this task. k-NN is a non-parametric method in which weights are calculated for classes for every feature. This non-parametric method is based on the distance for classification. J48 (C4.5) is a decision tree classifier that is a map of the possible classes. This map

Table 5. The Performance Results of Unsupervised, Supervised, and TC Approaches

Method	Acc	Rec	Pre	f-M	Log Loss	CT(s)	PT(s)
Methods in the Unsupervised Approaches*							
Bhardwaj and Mangat [6]	0.855	0.922	0.188	0.313	5.002	–	0.782
Helfman and Hollan [17]							
Largest IS	0.958	0.384	0.407	0.395	1.451	–	0.735
Largest W	0.937	0.582	0.301	0.396	2.184	–	0.682
Largest H	0.930	0.570	0.272	0.368	2.416	–	0.702
Highest W*H	0.931	0.590	0.281	0.380	2.373	–	0.652
Gali et al. [14]	0.577	0.843	0.067	0.125	14.615	–	0.755
Fazal et al. [10]	0.840	0.921	0.173	0.292	5.551	–	0.788
Machine Learning Methods in the Supervised Approaches**							
RBF SVM [41, 45]	0.974	0.378	0.671	0.484	0.893	668.192	282.455
k-NN [41]	0.952	0.652	0.341	0.416	1.655	5.859	15.852
J48 [41]	0.964	0.656	0.516	0.572	1.408	3.588	0.068
RF [41]	0.986	0.723	0.860	0.785	0.477	55.916	2.208
AdaBoost [41]	0.987	0.787	0.824	0.805	0.459	27.738	2.936
Machine Learning Methods in the TC Approach***							
RBF SVM	0.983	0.882	0.861	0.840	0.596	**0.011**	0.214
k-NN	0.990	0.891	0.948	0.900	0.329	0.020	0.070
J48	0.992	0.953	0.956	0.947	0.260	**0.011**	**0.001**
RF	**0.995**	0.954	**0.975**	**0.958**	**0.159**	0.121	0.030
AdaBoost	**0.995**	**0.957**	0.971	**0.958**	0.183	0.028	0.012

Acc: Accuracy, Rec: Recall, Pre: Precision, f-M: f-Measure, s: seconds.
W: Width, H, Height, IS: Image Size.
PT (Unsupervised, Supervised, TC): average prediction time per a website.
CT (Supervised): average construction time for 120 websites (12,000 web pages).
CT (TC): average construction time per website (six web pages).

is utilized for the prediction of classes and observables in the learning model. Moreover, we use two ensemble learning models to enhance the performance results of decision trees. RF constructs a multitude of decision trees to find the most appropriate tree for better prediction. AdaBoost method [33] combines weak classifiers (decision tree in this study) to obtain an even more accurate model.

5.4 Comparison of Performance Results on Unsupervised and Supervised Approaches with the TC Approach

In this section, we compare the average performance results of various methods that can be used in both unsupervised and supervised approaches, as well as our proposed novel approach. These results are the average of five tests, and the details of these tests will be explained in Sections 5.5 and 5.6. Additionally, we explore five machine learning methods to determine the most appropriate method for both supervised and TC approaches. Table 5 gives the performance results and average construction/prediction time results.

In the unsupervised approaches, we conducted tests on the entirety of the dataset, which included all 100 web pages from each of the 200 websites. According to Table 5, the f-Measure of a simple function (width > 300px and height > 400px) proposed by Bhardwaj and Mangat [6] is 0.313. Helfman and Hollan [17] select a single image, which is a large image on a web page, for finding a representative image of a web page. In our experiments, we derive four functions, including the

largest image size (IS), width (W), height (H), and W*H in a web page. The f-Measure results for the IS and the W of the image functions are 0.395 and 0.396, respectively, which are the best results among the unsupervised approaches. The function of selecting the highest pixel value (W * H) is the second-best result (0.380 f-Measure). Gali et al. [14] prepare a complex function (ratio < 1.8 and Width > 100px and Height > 100px and some ignored words including logo, banner, button, free, and so on.) for finding the representative images from web pages. However, the f-Measure of this function is 0.125, which is the worst result, because this complex function uses only English words. Fazal et al. [10] propose a simple function (width > 400px and height > 400px) for which the f-measure is 0.292. In these functions, the average prediction times for a website are very close to each other. Considering that there are 100 web pages on each website, there is a time delay of approximately 0.7 second for each web page. In these approaches, it is observed that the Log Loss values are considerably higher than 0, which indicates that the unsupervised approaches are not effective for this task. Therefore, the results of f-Measure and Log Loss highlight the inadequacy of unsupervised methods.

Table 5 shows that the results of supervised approaches are significantly better than those of unsupervised approaches. In supervised approaches, the size of the training dataset is an important parameter in building an accurate learning model. Therefore, the optimal training dataset size is examined in Section 5.6. We conducted supervised experiments using a 30-feature dataset, where 60% of the data was assigned as the training dataset and 40% as the testing dataset. The dataset was shuffled, and this process was repeated five times to construct different training and testing sets. In Table 5, when the average results are examined in terms of accuracy, it is seen that these results are too close. However, the number of relevant images on a web page is lower than the number of irrelevant images. Therefore, f-Measure is a much more appropriate metric for an imbalanced dataset in terms of interpreting the results. k-NN is the worst supervised method when compared to the others. The result of k-NN depends on the characteristics of the dataset. RBF SVM is dramatically the second-worst supervised method, although it has been suggested by Reference [45]. Even though the decision tree method has been suggested in many web extraction studies [36, 38, 42] in the literature, it is not sufficient for discovering relevant images. In this situation, ensemble learning models suggested in the literature can be used as a solution. In this study, we used two ensemble-based methods, including AdaBoost and RF, to improve the results. Previous studies on web data extraction have proved that AdaBoost [40, 41] and RF [52] give successful results in different imbalanced datasets. Furthermore, the ensemble-based methods were able to achieve Log Loss values below 0.5. This indicates that the learning models of these methods outperform others. As a result, ensemble-based methods provide significant improvements for this task.

In the supervised approaches, a certain amount of time is required to construct a learning model using the training dataset and to predict relevant or irrelevant images. Table 5 gives the model construction time and prediction time for 12,000 web pages obtained from 120 websites and 80 websites that contain 8,000 web pages, respectively. The decision tree method achieves the best time results for both construction and prediction times. However, the AdaBoost method achieves the best performance results when considering both the f-Measure and time results. While this method creates the learning model on the training dataset in 27.738 seconds, the average prediction time is 2.936 seconds per website.

In the TC approach, only textual data is selected from the dataset, and for each website, 6 web pages were used for training while 94 web pages were used for testing. The tests were randomly generated five times for 200 websites. In this approach, a non-expert user annotates only the relevant images of several web pages on a website for constructing the training dataset. In the analysis presented in Section 5.5, it has been observed that the success of the prediction model has reached satisfactory results in 6 web pages. For this reason, the average results given in Table 5 use 6 web

pages as a training dataset on a website. Then, the performance of relevant image prediction is tested on 94 web pages of the same website. Many supervised methods can be tried on the machine learning step of the TC approach. Table 5 gives the experimental results of these methods. Most existing approaches to predicting images on web pages require large amounts of training data and time. In contrast, the TC approach achieves high accuracy with only a small training dataset. Our approach can involve ensemble-based methods, which are effective in both supervised and TC approaches. Specifically, the Adaboost and RF methods perform best in terms of f-Measure and Log Loss values. In addition, our ensemble methods achieve an f-Measure of around 0.958 for 6 web pages. When considering the time required for model construction and prediction tasks, the Adaboost method is more efficient than the RF method. By using our approach, only 1,200 web pages are needed for training to predict images on 18,800 web pages across 200 websites. This is a significant improvement compared to the supervised approach, which requires training with 12,000 web pages. Overall, our approach offers a more efficient and accurate solution for relevant image prediction on web pages.

When considering time efficiency, the decision tree method stands out in the TC approach, with an average prediction time of around 0.001 second for 94 web pages across 200 websites. The decision tree method also has an average construction time of 0.011 second, which is comparable to the AdaBoost and RF methods, achieving an f-Measure of 0.947. The AdaBoost method follows closely with an average construction time of 0.028 second and an average prediction time of 0.012 second. Our approach improves both construction and prediction times compared to the TC approach, as shown in Table 5. However, downloading images is the most time-consuming operation, taking about 0.510 second, as shown in Table 4. Thankfully, our approach does not require this costly operation when predicting images.

5.5 Benefit of the Clustering Step: Investigating the Impact of Training Dataset Size and Number of URLs

In this section, a sequential examination of two crucial factors affecting the performance of the clustering algorithm is conducted: the size of the training dataset and the optimal number of URLs. The analysis progresses in stages, beginning with the exploration of different training dataset sizes. Clustering results obtained from varying training dataset sizes are compared and evaluated. Subsequently, the focus shifts to identifying the minimum number of URLs necessary for reliable clustering outcomes. By adopting this step-by-step approach, the goal is to systematically investigate the efficiency and effectiveness of the clustering algorithm of our approach.

In the first stage, to compare the proposed clustering step of the TC approach in constructing training datasets for each website, the TC approach is compared with two different strategies. The first strategy involves selecting a URL from the clusters returned by DBSCAN, and the values obtained from the first loop in Algorithm 1 are used as the training data. The second strategy involves randomly selecting a URL from the available N URLs. The approach proposed in this study is a combination of these two strategies. Table 6 shows a comparison of the f-Measures obtained using these three strategies. To evaluate whether the results are significant, the average f-Measures are calculated for 200 websites, and the experiments are repeated five times as the selection process is random. Table 6 presents the average f-Measures of the five experiments for each condition. Additionally, the f-Measures obtained using the AdaBoost method, which yields successful results in the experiments, is used in this experiment.

Table 6 indicates that when only clustering is used with URLs returned from DBSCAN, 2–4 clusters are obtained. After downloading the web page of the URL in these clusters and annotating the related images, the learning model obtained from the AdaBoost method was tested on the web pages. For example, 3 web pages were used for the DBSCAN 3 clustering results out of 100 pages

Table 6. Comparison of f-Measures Obtained from Training Datasets Created by Random Selection (Non-clustering), Only with Clustering and the Clustering Step of TC

	Only with Cluster	
	f-Measure ± Std.	Size of Clusters ± Std.
	0.664 ± 0.410	2.607 ± 1.615

Number of URLs *trainingSize*	**Random Selection** f-Measure ± Std.	**Clustering Step of TC** f-Measure ± Std.
1	0.483 ± 0.426	
2	**0.806 ± 0.280***	0.676 ± 0.408
3	0.889 ± 0.199	0.876 ± 0.233
4	0.915 ± 0.156	**0.927 ± 0.153***
5	0.921 ± 0.142	**0.947 ± 0.120***
6	0.941 ± 0,118	**0,958 ± 0,093***
7	0.947 ± 0,116	**0,959 ± 0,098***
8	0.950 ± 0,106	**0,965 ± 0,081***
9	0.950 ± 0,109	**0,970 ± 0,073***
10	0.958 ± 0,093	**0,970 ± 0,074***
11	0.959 ± 0,092	**0,974 ± 0,070***
12	0.967 ± 0,082	**0,975 ± 0,065***
13	0.968 ± 0,082	**0,975 ± 0,064***
14	0.967 ± 0,084	**0,977 ± 0,063***
15	0.969 ± 0,082	**0,980 ± 0,051***

*statistical significance for $p < 0.050$.

of a website, and the remaining 97 pages were used for evaluating the performance of the learning model. With this strategy, an average f-Measure value of 0.664 was obtained for 200 websites. However, it was observed that selecting only one URL from clusters containing too much data is insufficient, because the training data would be formed improperly if there were no images on the web page of the selected URL.

In the second strategy, a learning model is generated by randomly selecting a certain number of URLs from 100 web pages and using them as the testing dataset for the remaining pages. In the experiments, training models are created based on the textual data of the images returned from web pages consisting of between 1 and 15 URLs. When only 1 URL was used, the average f-Measure for five tests on 200 websites is 0.483. As the number of URLs used for the training dataset increases, the f-Measure improves and the standard deviation decreases.

Similar experiments are conducted for the clustering step, which is one of the important steps of the TC approach. At least 2 URLs are required for the clustering step. When there are two clusters, 2 URLs are randomly selected from the clusters, regardless of the number of clusters. In other words, although DBSCAN returns three clusters, 2 URLs are chosen so the tests are conducted under equal conditions. An average f-Measure of 0.806 is obtained in random selection for the two clusters. In the clustering step of the TC approach, an average f-Measure of 0.676 is obtained. For 2 URLs, random selection is significantly better than the clustering step in our approach. However, using only 2 URLs to represent a website in the training dataset is not sufficient. When there are 3 URLs in the training dataset, there is no significant difference in performance between the two strategies. When the number of URLs is 4 or more, the TC approach is significantly better than random selection. The TC approach achieves an average f-Measure value of 0.958 on 200

Table 7. Scaling Analysis of Various Size of URLs for the Clustering Algorithm

Size of URLs	f-Measure ± Std.
10	0.939 ± 0.143
20	0.942 ± 0.144
30	0.951 ± 0.128
40	**0.953 ± 0.126**
50	**0.955 ± 0.113**
60	**0.958 ± 0.112**
70	**0.959 ± 0.106**
80	**0.958 ± 0.109**
90	**0.958 ± 0.116**
100	**0.958 ± 0.093**

*statistical significance for $p < 0.050$.

websites for the learning model established by annotating relevant images on 6 web pages for a website by a user.

In the second stage of this section, the aim is to determine the minimum number of URLs required to select a set of 6 web pages for training. The investigation involves systematically varying the number of URLs in the dataset. Starting with 10 URLs, the dataset size is incrementally increased by 10 up to 100 URLs. For each website, the clustering algorithm is utilized to select 6 web pages for training. The remaining 94 web pages are then designated for testing. The experiment is repeated five times to ensure the robustness of the results. The f-Measure values, which assess the effectiveness of the clustering model, are presented in Table 7. Additionally, the statistical significance of the observed changes in the f-Measure values as the number of URLs increases is examined.

Table 7 presents the average performance results of the learning model based on a varying number of URLs for the clustering step. The model achieves an average f-Measure value of 0.939 for 200 websites when trained with 6 pages selected from 10 URLs. As the number of URLs increases, the f-Measure value demonstrates improvement, reaching 0.958 with 100 URLs. Notably, after using 40 URLs, the obtained values show significant similarity to those obtained with 100 URLs. Moreover, the f-Measure values for 60 URLs and beyond exhibit consistent performance. Based on these findings, it can be suggested that 60 URLs may be considered an optimal value. This section investigates the determination of optimal values; however, it should be noted that these values may vary across different websites.

5.6 Investigating the Influence of Training Dataset Size in Supervised Approaches and the Effect of Using Textual Data in These Approaches

In the final experiment, the aim was to determine the optimal size of the training dataset for the supervised approaches. For this purpose, we used the AdaBoost method, which provided the best f-Measure results in the evaluations. To conduct this experiment, a certain number of websites were selected for the training dataset, while the remaining websites were used for the testing dataset. For instance, in the first iteration, the learning model was constructed from 20 websites, and then it was evaluated on 180 websites. In the second iteration, 40 websites were used for creating the learning model, and 160 websites were used for testing this model. This process was repeated, adding 20 websites in each iteration. The tests were repeated five times, and the websites were shuffled each time to avoid any bias.

Table 8. The Impacts of Training Dataset Size in the Prediction Performance of Supervised Approaches

Number of Websites for Training	**Dataset 1** f-Measure ± Std.	**Dataset 2** f-Measure ± Std.
20	**0.659 ± 0.105***	0.195 ± 0.046
40	**0.749 ± 0.030***	0.249 ± 0.032
60	**0.763 ± 0.008***	0.264 ± 0.042
80	**0.790 ± 0.018***	0.272 ± 0.043
100	**0.805 ± 0.019***	0.289 ± 0.036
120*	**0.805 ± 0,016***	0,304 ± 0,044
140	**0.800 ± 0,018***	0,293 ± 0,080
160	**0.797 ± 0,020***	0,271 ± 0,077

*statistical significance for $p < 0.050$.
Dataset-1: 30 features, Dataset-2: Only Textual Data.

Table 8 shows the mean results and standard deviations of these tests for two different datasets. The first dataset (Dataset-1), suggested by Uzun el al. [41], contains 30 features. The second dataset (Dataset-2) represents only textual data obtained from images and their parent tags, similar to our approach. The table shows the number of websites used as the training dataset and the f-Measures obtained from the prediction of images on the remaining websites as the testing dataset.

In Table 8, the first iteration using the supervised approach with 10,000 web pages downloaded from 10 websites resulted in an f-Measure of 0.659. However, when using only textual data to create the learning model, the f-Measure was very low at 0.195. Looking at all the f-Measures in Table 8, it becomes clear that the learning model obtained from textual data is not sufficient for a general solution, as web developers use different layouts, tags, attributes, and elements for their websites. Therefore, the textual data used to create the images varies from website to website.

Furthermore, the f-Measure of supervised approaches improves with an increase in the size of the training dataset. The best f-Measure of 0.805 was obtained using a training dataset consisting of 100/120 websites and a testing dataset containing 100/120 websites, respectively. However, no significant change was observed in the f-Measure beyond the training dataset size of 100. In Experiment 5.4, we used 120 websites, because the standard deviation was slightly better.

5.7 Discussion

Web scraping tools use manual extraction approaches to extract desired HTML elements from web pages. However, these tools typically require a certain level of web development knowledge to properly set up and use. They are typically useful for extracting data from a single website, because the extraction expressions used in manual extraction can vary between websites and even between different layouts of the same website. In contrast, supervised approaches can be difficult to implement in web scraping tools. The TC approach can be a viable alternative to manual extraction approaches in web scrapers, as it can be more easily integrated into these tools.

Semantic web technologies act as a bridge to transform unstructured web data into structured data. That is, these technologies enable people to write their own rules for easily scraping data. The schema.org community specifies schemas for obtaining structured data. However, most web developers do not apply these schemas to their websites. Some developers only use these schemas for their own needs. As a result, the desired development in the semantic web has not been achieved yet, despite years of attempts. Automatic web scraping studies can enhance the semantic web creation task. Our study contributes to the determination of relevant images from web pages on a website.

This study evaluates the performance of the TC approach against various unsupervised and supervised approaches for the task of extracting relevant images from web pages. Unsupervised approaches perform poorly, with the highest f-Measure of 0.396, and their scoring technique is not suitable for imbalanced datasets like this study. The last study [19] in unsupervised approaches calculates a score for each image on a web page and selects the first three images as relevant. Given that there are approximately 1.285 relevant images for a web page in our dataset, 1.715 images are being misclassified as relevant using this technique. So, this scoring technique is not suitable for our dataset. In the experiments, supervised approaches yield significantly better performance compared to unsupervised approaches. Among the supervised approaches, the f-Measure for the RBF SVM method proposed by Reference [45] is 0.439 in their dataset, while the f-Measure in our dataset is 0.484. Compared to other supervised methods, this method is not suitable for an imbalanced dataset. In this study, the AdaBoost method, which is one of the ensemble methods, gives better performance (0.805 f-Measure with 60% of the dataset as training and 0.822 f-Measure [41] with 75% of the dataset as training) for this task. There are several crucial tasks, including discovering proper features and gathering/annotating a lot of data for creating an automatic extraction system. However, these tasks are time-consuming. Therefore, the manual extraction approaches are preferred by web scraping tools. However, these approaches depend on the experience of the user. Furthermore, the task of examining many web pages on a website is cumbersome. Therefore, we propose a semi-automatic approach that focuses on the website-based and only the textual data of the HTML elements. Our approach builds a learning model for a website using a few annotations, achieving an f-Measure of 0.958 by annotating only six web pages for a website, thereby automating the task of the expert user in manual extraction approaches.

This study introduces a novel step for identifying web layouts by clustering URLs obtained from a small number of web pages. Traditionally, web layout identification methods focus [15, 28, 34] on individual web pages, but our approach considers the URLs collectively. By randomly selecting six URLs and creating a training dataset, we were able to achieve an f-Measure of 0.941 for the learning model, which was significantly improved by our proposed clustering step and reached an f-Measure of 0.958.

In the literature on web extraction, researchers have been looking for general solutions by considering and studying different features obtained from HTML elements. However, textual data, which is one of the features, was not taken into account in the existing approaches, since this feature is not suitable for a general solution. Experiment 5.6 in this study indicates that using only textual data yielded a very low f-measure of 0.304 in supervised approaches. However, our proposed approach demonstrates that the textual data can be used in the web data scraping task based on the website, resulting in a more successful approach.

Extracting relevant images from web pages using supervised approaches often involves time-consuming operations such as downloading images and preparing features for prediction. Typically, the most important features for image prediction are obtained by downloading the image, which takes an average of 0.510 second per web page. However, our TC approach eliminates the need for downloading the image by relying solely on the textual data associated with the image. Our approach can extract only the source of the image, without storing the actual image files. As demonstrated in Section 5.1, our approach significantly reduces the amount of time and storage space required for relevant image extraction compared to previous approaches.

6 CONCLUSION

In the field of web data extraction, previous academic studies have focused on developing generalized learning models (unsupervised and supervised approaches) that can be applied to scrape data from various web pages. However, accurately extracting relevant images from web pages presents

a challenging issue in this process, requiring significant resources and execution time. As a result, web experts do not typically rely on generalized solutions for this task, preferring instead to generate patterns by manually examining the textual data of web pages on a given website. While web scraping tools support manual extraction approaches that utilize patterns, these approaches are time-consuming and require significant expertise. In this study, we propose a semi-automatic approach that creates a learning model based on the textual data of web pages from a single website, eliminating the need for expert input. Our approach demonstrates significantly improved performance compared with generalized solutions, both in terms of f-Measure and execution time. Additionally, our approach is easily integrated into web scraping tools, since it relies on the textual data that these tools are designed to handle. Overall, this study highlights the potential benefits of using a semi-automatic approach for relevant image extraction, particularly for websites with a large number of web pages.

Traditionally, unsupervised and supervised approaches to relevant image extraction have not utilized textual data, because they are not deemed useful for prediction models. However, this shows that textual data can be effectively utilized in such approaches. Moreover, the study demonstrated that a successful learning model can be established using only a few annotated images related to a website. By clustering URLs, the training data construction task can be made more efficient, requiring the user to examine fewer web pages.

In future studies, we will focus on four different areas. First, we plan to apply the approach developed in this study to other web extraction tasks. Second, we aim to determine the most suitable regular expressions to scrape data from web pages by analyzing both positive and negative texts of HTML elements. Our third objective is to generalize the solution obtained from the learning models constructed for websites. Finally, we intend to examine the clustering process of our approach for more complex layouts and improve it accordingly.

REFERENCES

[1] Hayri Volkan Agun and Erdinç Uzun. 2023. An efficient regular expression inference approach for relevant image extraction. *Appl. Soft Comput.* 135 (2023), 110030. DOI: https://doi.org/10.1016/j.asoc.2023.110030

[2] Julian Alarte, David Insa, Josep Silva, and Salvador Tamarit. 2018. Main content extraction from heterogeneous webpages. In *Web Information Systems Engineering (WISE'18)*, Hakim Hacid, Wojciech Cellary, Hua Wang, Hye-Young Paik, and Rui Zhou (Eds.). Springer International Publishing, Cham, 393–407.

[3] Naseer Aslam, Bilal Tahir, Hafiz Muhammad Shafiq, and Muhammad Amir Mehmood. 2019. Web-AM: An efficient boilerplate removal algorithm for web articles. In *International Conference on Frontiers of Information Technology (FIT'19)*. IEEE, 287–2875. DOI: https://doi.org/10.1109/FIT47737.2019.00061

[4] Ziv Bar-Yossef and Sridhar Rajagopalan. 2002. Template detection via data mining and its applications. In *11th International Conference on World Wide Web (WWW'02)*. Association for Computing Machinery, New York, NY, 580–591. DOI: https://doi.org/10.1145/511446.511522

[5] Rodrigo Barbado, Oscar Araque, and Carlos A. Iglesias. 2019. A framework for fake review detection in online consumer electronics retailers. *Inf. Process. Manag.* 56, 4 (2019), 1234–1244. DOI: https://doi.org/10.1016/j.ipm.2019.03.002

[6] Aanshi Bhardwaj and Veenu Mangat. 2014. An improvised algorithm for relevant content extraction from web pages. *J. Emerg. Technol. Web Intell.* 6, 2 (May 2014), 226–230. DOI: https://doi.org/10.4304/jetwi.6.2.226-230

[7] Lidong Bing, Tak-Lam Wong, and Wai Lam. 2016. Unsupervised extraction of popular product attributes from e-commerce web sites by considering customer reviews. *ACM Trans. Internet Technol.* 16, 2, Article 12 (Apr. 2016), 17 pages. DOI: https://doi.org/10.1145/2857054

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*. AAAI Press, 226–231.

[9] Fadwa Estuka and James Miller. 2019. A pure visual approach for automatically extracting and aligning structured web data. *ACM Trans. Internet Technol.* 19, 4, Article 51 (Nov. 2019), 26 pages. DOI: https://doi.org/10.1145/3365376

[10] Nancy Fazal, Khue Nguyen, and Pasi Fränti. 2019. Efficiency of web crawling for geotagged image retrieval. *Webology* 16 (2019), 16–39. DOI: https://doi.org/10.14704/WEB/V16I1/a177

[11] Emilio Ferrara and Robert Baumgartner. 2011. Automatic wrapper adaptation by tree edit distance matching. In *Combinations of Intelligent Methods and Applications*. Springer, UK, 41–54.

[12] Leandro Neiva Lopes Figueiredo, Guilherme Tavares de Assis, and Anderson A. Ferreira. 2017. DERIN: A data extraction method based on rendering information and n-gram. *Inf. Process. Manag.* 53, 5 (2017), 1120–1138. DOI: https://doi.org/10.1016/j.ipm.2017.04.007

[13] Jeffrey E. F. Friedl and Andy Oram. 2002. *Mastering Regular Expressions* (2nd ed.). O'Reilly & Associates, Inc.

[14] Najlah Gali, Andrei Tabarcea, and Pasi Fränti. 2015. Extracting representative image from web page. In *11th International Conference on Web Information Systems and Technologies (WEBIST'15)*. INSTICC, SciTePress, Portugal, 411–419. DOI: https://doi.org/10.5220/0005438704110419

[15] Waqar Haider and Yeliz Yesilada. 2022. Classification of layout vs. relational tables on the web: Machine learning with rendered pages. *ACM Trans. Web* 17, 1, Article 1 (Dec. 2022), 23 pages. DOI: https://doi.org/10.1145/3555349

[16] Wook-Shin Han, Wooseong Kwak, Hwanjo Yu, Jeong-Hoon Lee, and Min-Soo Kim. 2014. Leveraging spatial join for robust tuple extraction from web pages. *Inf. Sci.* 261 (2014), 132–148. DOI: https://doi.org/10.1016/j.ins.2013.09.027

[17] Jonathan I. Helfman and James D. Hollan. 2000. Image representations for accessing and organizing web information. In *Internet Imaging II*, Giordano B. Beretta and Raimondo Schettini (Eds.), Vol. 4311. International Society for Optics and Photonics, SPIE, San Jose, CA, 91–101. DOI: https://doi.org/10.1117/12.411880

[18] Chun-Nan Hsu and Ming-Tzung Dung. 1998. Generating finite-state transducers for semi-structured data extraction from the web. *Inf. Syst.* 23, 8 (1998), 521–538. DOI: https://doi.org/10.1016/S0306-4379(98)00027-1

[19] Imranul Islam. 2021. *Representative Image Extraction from Web Page*. Master's Thesis. University of Eastern Finland, Faculty of Science and Forestry, Joensuu School of Computing.

[20] Patricia Jiménez, Juan C. Roldán, and Rafael Corchuelo. 2021. A clustering approach to extract data from HTML tables. *Inf. Process. Manag.* 58, 6 (2021), 102683. DOI: https://doi.org/10.1016/j.ipm.2021.102683

[21] Yeonjung Kim, Jeahyun Park, Taehwan Kim, and Joongmin Choi. 2007. Web information extraction by HTML tree edit distance matching. In *International Conference on Convergence Information Technology (ICCIT'07)*. IEEE, 2455–2460. DOI: https://doi.org/10.1109/ICCIT.2007.19

[22] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *3rd ACM International Conference on Web Search and Data Mining (WSDM'10)*. Association for Computing Machinery, New York, NY, 441–450. DOI: https://doi.org/10.1145/1718487.1718542

[23] Vladimir Iosifovich Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl.* 10, 8 (1966), 707–710.

[24] Bing Liu. 2011. *Web Data Mining Exploring Hyperlinks, Contents, and Usage Data*. Springer, Berlin. DOI: https://doi.org/10.1007/978-3-642-19460-3

[25] Qingtang Liu, Mingbo Shao, Linjing Wu, Gang Zhao, Guilin Fan, and Jun Li. 2017. Main content extraction from web pages based on node characteristics. *J. Comput. Sci. Eng.* 11 (06 2017), 39–48. DOI: https://doi.org/10.5626/JCSE.2017.11.2.39

[26] Pedro Lopez-Garcia, Antonio D. Masegosa, Eneko Osaba, Enrique Onieva, and Asier Perallos. 2019. Ensemble classification for imbalanced data based on feature space partitioning and hybrid metaheuristics. *Appl. Intell.* 49, 8 (Aug. 2019), 2807–2822. DOI: https://doi.org/10.1007/s10489-019-01423-6

[27] Mehul Mahrishi, Sudha Morwal, Nidhi Dahiya, and Hanisha Nankani. 2021. A framework for index point detection using effective title extraction from video thumbnails. *Int. J. Syst. Assur. Eng. Manag.* (June 2021), 1–6. DOI: https://doi.org/10.1007/s13198-021-01166-z

[28] Edimar Manica, Carina Friedrich Dorneles, and Renata Galante. 2019. Combining URL and HTML features for entity discovery in the web. *ACM Trans. Web* 13, 4, Article 20 (Dec. 2019), 27 pages. DOI: https://doi.org/10.1145/3365574

[29] Ion Muslea, Steve Minton, and Craig Knoblock. 1999. A hierarchical approach to wrapper induction. In *3rd Annual Conference on Autonomous Agents (AGENTS'99)*. Association for Computing Machinery, New York, NY, 190–197. DOI: https://doi.org/10.1145/301136.301191

[30] D. C. Reis, P. B. Golgher, A. S. Silva, and A. F. Laender. 2004. Automatic web news extraction using tree edit distance. In *13th International Conference on World Wide Web (WWW'04)*. Association for Computing Machinery, New York, NY, 502–511. DOI: https://doi.org/10.1145/988672.988740

[31] Arnaud Sahuguet and Fabien Azavant. 1999. Building light-weight wrappers for legacy web data-sources using W4F. In *25th International Conference on Very Large Data Bases (VLDB'99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 738–741.

[32] Roland Schäfer. 2017. Accurate and efficient general-purpose boilerplate detection for crawled web corpora. *Lang. Resour. Eval.* 51 (2017), 873–889.

[33] Robert E. Schapire and Yoav Freund. 2012. *Boosting: Foundations and Algorithms*. The MIT Press, London, England.

[34] Andrés Soto, Héctor Mora, and Jaime A. Riascos. 2022. Web generator: An open-source software for synthetic web-based user interface dataset generation. *SoftwareX* 17 (2022), 100985. DOI: https://doi.org/10.1016/j.softx.2022.100985

[35] Erdinç Uzun. 2020. A regular expression generator based on CSS selectors for efficient extraction from HTML pages. *Turk. J. Electric. Eng. Comput. Sci.* 28 (2020), 3389–3401. DOI: https://doi.org/10.3906/elk-2004-67

[36] Erdinç Uzun, Hayri Volkan Agun, and Tarik Yerlikaya. 2013. A hybrid approach for extracting informative content from web pages. *Inf. Process. Manag.* 49, 4 (2013), 928–944.

[37] Erdinç Uzun, Halil Nusret Buluş, Alpay Doruk, and Erkan Özhan. 2017. Evaluation of HAP, AngleSharp and HtmlDocument in web content extraction. In *International Scientific Conference (UNITECH'17)*. UNITECH, 275–278.

[38] Erdinç Uzun, Edip Serdar Güner, Yılmaz Kılıçaslan, Tarık Yerlikaya, and Hayri Volkan Agun. 2014. An effective and efficient web content extractor for optimizing the crawling process. *Softw.: Pract. Exper.* 44, 10 (2014), 1181–1199. DOI: https://doi.org/10.1002/spe.2195

[39] Erdinç Uzun, Tarık Yerlikaya, and Oğuz Kırat. 2018. Comparison of Python libraries used for web data extraction. *J. Technic. Univ. - Sofia Plovdiv branch, Bulgar.* 24 (2018), 87–92.

[40] Erdinç Uzun and Erkan Özhan. 2018. Examining the impact of feature selection on classification of user reviews in web pages. In *International Conference on Artificial Intelligence and Data Processing (IDAP'18)*. IEEE, 1–8. DOI: https://doi.org/10.1109/IDAP.2018.8620774

[41] Erdinç Uzun, Erkan Özhan, Hayri Volkan Agun, Tarik Yerlikaya, and Halil Nusret Buluş. 2020. Automatically discovering relevant images from web pages. *IEEE Access* 8 (2020), 208910–208921. DOI: https://doi.org/10.1109/ACCESS.2020.3039044

[42] Erdem Uçar, Erdinç Uzun, and Pınar Tüfekci. 2017. A novel algorithm for extracting the user reviews from web pages. *J. Inf. Sci.* 43, 5 (2017), 696–712. DOI: https://doi.org/10.1177/0165551516666446

[43] Dinesh Kumar Vishwakarma, Deepika Varshney, and Ashima Yadav. 2019. Detection and veracity analysis of fake news via scrapping and authenticating the web search. *Cognit. Syst. Res.* 58 (2019), 217–229. DOI: https://doi.org/10.1016/j.cogsys.2019.07.004

[44] Thijs Vogels, Octavian-Eugen Ganea, and Carsten Eickhoff. 2018. Web2Text: Deep structured boilerplate removal. In *Advances in Information Retrieval*, Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury (Eds.). Springer International Publishing, Cham, 167–179.

[45] Krishna Vyas and Flavius Frasincar. 2020. Determining the most representative image on a web page. *Inf. Sci.* 512 (2020), 1234–1248. DOI: https://doi.org/10.1016/j.ins.2019.10.045

[46] Yu-Chieh Wu. 2016. Language independent web news extraction system based on text detection framework. *Inf. Sci.* 342 (2016), 132–149. DOI: https://doi.org/10.1016/j.ins.2015.12.025

[47] Junxiao Xue, Yabo Wang, Yichen Tian, Yafei Li, Lei Shi, and Lin Wei. 2021. Detecting fake news by exploring the consistency of multimodal data. *Inf. Process. Manag.* 58, 5 (2021), 102610. DOI: https://doi.org/10.1016/j.ipm.2021.102610

[48] Yanhong Zhai and Bing Liu. 2006. Structured data extraction from the web based on partial tree alignment. *IEEE Trans. Knowl. Data Eng.* 18, 12 (2006), 1614–1628. DOI: https://doi.org/10.1109/TKDE.2006.197

[49] Shuo Zhang and Krisztian Balog. 2020. Web table extraction, retrieval, and augmentation: A survey. *ACM Trans. Intell. Syst. Technol.* 11, 2, Article 13 (Jan. 2020), 35 pages. DOI: https://doi.org/10.1145/3372117

[50] Huiliang Zhao, Zhenghong Liu, Xuemei Yao, and Qin Yang. 2021. A machine learning-based sentiment analysis of online product reviews with a novel term weighting and feature selection approach. *Inf. Process. Manag.* 58, 5 (2021), 102656. DOI: https://doi.org/10.1016/j.ipm.2021.102656

[51] Ziyuan Zhao, Huiying Zhu, Zehao Xue, Zhao Liu, Jing Tian, Matthew Chin Heng Chua, and Maofu Liu. 2019. An image-text consistency driven multimodal sentiment analysis approach for social media. *Inf. Process. Manag.* 56, 6 (2019), 102097. DOI: https://doi.org/10.1016/j.ipm.2019.102097

[52] Erkan Özhan and Erdinç Uzun. 2018. Performance evaluation of classification methods in layout prediction of web pages. In *International Conference on Artificial Intelligence and Data Processing (IDAP'18)*. IEEE, 1–7. DOI: https://doi.org/10.1109/IDAP.2018.8620893

Received 11 August 2022; revised 1 June 2023; accepted 2 August 2023